INTRAMODULE MULTIELEMENT MICROSYSTEM (IM^2) BUS AND INTERFACE CIRCUIT

by

Zhigang Wang

A report submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Electrical and Computer Engineering)
in the University of Kentucky
July, 2001

Committee in charge:

Assistant Professor Andrew Mason, Chairperson
Assistant Professor Zhi Chen
Associate Professor J. Robert Heath

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
01-07-2001	Thesis	xx-xx-2001 to xx-xx-2001

4. TITLE AND SUBTITLE
Intramodule Multielement Microsystem (IM2) Bus and Interface Circuit
Unclassified

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Wang, Zhigang ;

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
University of Kentucky
xxxxx
xxxxx, KYxxxxx

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
,

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes a communication bus for multielement "smart sensor" microsystems and the architecture for a sensor bus interface circuit. The Intramodule Multielement Microsystem (IM2) Bus which has been developed provides a link between central control electronics and individual front-end transducers in a sensor-based microsystem. The Transducer Independent Interface (TII) of the IEEE standard for a Smart Transducer Interface for Sensors and Actuators (IEEE 1451.2) has been adopted and expanded in this implementation to support multi-node digital communication. Modifications include signals, protocols, and architecture. The microsystem interface circuit which has been designed includes an IM2 Bus interface, 128bits SRAM, 8 bits of multifunction I/O, a boundary-scan test chain, and an SPI bus interface to support an external EEPROM, a serial A/D and/or a D/A. Its advanced features include online control of sensor readout such as sensor range selection, amplifier gain and offset control, interrupt parameter threshold selection, automatic network configuration (plug-n-play features), and online self-test. It owns real time interrupt and controllable power supplies to keep system working in very low power consumption level. A user interface to this IC for test and development, which consists of both hardware and software, is also presented.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Public Release	53	Fenster, Lynn lfenster@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

INTRAMODULE MULTIELEMENT MICROSYSTEM (IM^2) BUS AND INTERFACE CIRCUIT

by

Zhigang Wang

ABSTRCT

This report describes a communication bus for multielement "smart sensor" microsystems and the architecture for a sensor bus interface circuit. The Intramodule Multielement Microsystem (IM^2) Bus which has been developed provides a link between central control electronics and individual front-end transducers in a sensor-based microsystem. The Transducer Independent Interface (TII) of the IEEE standard for a Smart Transducer Interface for Sensors and Actuators (IEEE 1451.2) has been adopted and expanded in this implementation to support multi-node digital communication. Modifications include signals, protocols, and architecture. The microsystem interface circuit which has been designed includes an IM^2 Bus interface, 128bits SRAM, 8 bits of multifunction I/O, a boundary-scan test chain, and an SPI bus interface to support an external EEPROM, a serial A/D and/or a D/A. Its advanced features include online control of sensor readout such as sensor range selection, amplifier gain and offset control, interrupt parameter threshold selection, automatic network configuration (plug-n-play features), and online self-test. It owns real time interrupt and controllable power supplies to keep system working in very low power consumption level. A user interface to this IC for test and development, which consists of both hardware and software, is also presented.

ACKNOWLEDGMENTS

I would like to express my great gratitude to Dr. Andrew Mason for his guidance and support throughout my graduate study and this Master project. Thanks for his always having that good and encouraging piece of advice ready. Thanks for his visions and leadership in my project.

I would like to thank Kun, Arivand, Jichun, YeGu and all members of AMSAC lab with whom I have collaborated over the past two years.

I would like to thank Dr. Chen and Dr. Heath for their services in my committee. I appreciate their valuable contributions.

TABLE OF CONTENTS

Chapter 1
Introduction

1.1. Smart Sensor and Microsystems

The market for sensors and sensor-based systems has become very large and continues to grow. The need for inexpensive, networkable sensors that are simple to use is tremendous, and would grow significantly if there were improvements in the cost-effective production of so-called "smart sensors" which provide intelligence at the sensor node rather than relying completely on the system for control and signal conditioning. Utilizing smart sensors will decentralize decisions and minimize logistics. Moreover, many upcoming sensor applications will require measurements of multiple parameters creating a demand for sensors that work in a network environment.

It is difficult to precisely define a smart sensor. Some organizations apply the label to any sensor that generates digital output. However, a truly smart sensor must do something with the data, not just pass them on in a digital format. A smart sensor must be able to not only collect and manage sensor data, but also to interact with the sensor front end (through digital communication) to implement some advanced features. It should respond to input signals in a logical fashion to increase the value of the information that it processes. A smart sensor should be capable of making logical decisions at the source of the information. It should support online sensor operation control, like sensor range selection, sensor readout control (gain and offset) and parameter threshold selection. A truly smart sensor can reduce system maintenance costs significantly by self-identification, plug-n-play capabilities, and reduction in cabling cost.

The development of microsystems, miniature low-power electronics capable of interacting with the physical world, has closely paralleled the growth of smart sensors. This is largely because microsystems provide a vehicle for implementing smart sensors in more useful applications and because many microsystems rely largely on smart sensors for their interaction with the physical world. In fact, the distinction between microsystems and smart sensors is often blurred in the literature and the two terms are often used interchangeably. However, for this research, a

microsystem will be defined as a system of components which may include one or more smart sensors and is therefore "greater" than an individual smart sensor. Thus, microsystems typically provide significantly more capability than a smart sensor, but generally utilize smart sensors to achieve their design goals.

Given the advantages of the smart sensor, the network-compatible smart sensors should be widely used. However the traditional signal chain, composed of sensor, signal conditioning, and A/D conversion, has proven difficult to replace. For instance, various devices have recently emerged that combine a number of links in a single device, such as the micromachined pressure sensors and accelerometers that incorporate signal conditioning to present a high-level (e.g., 0-5V) output. It throws in A/D as well, delivering a digitized output (parallel I/O or pulse train) for direct connection to a microcontroller or DSP. Although the device is quite appealing, the problem in designing systems with this component is that there is little agreement in the industry on digital I/O format and even less on the software machinations required to access the sensor data. The result is that a seemingly simple change from brand *x* to brand *y* is likely to require a wholesale redesign of both hardware and software [6].

Another problem is that most of sensors on the market only communicate through basic point-to-point usage. This may work fine for only a few channels, but it creates a tremendous wiring overhead which can only be minimized by connecting multiple devices on a single digital bus. An alternative choice is the so-called *fieldbus* concept which provides a single bus with multiple nodes. However, fieldbus is designed for communication between microcontrollers and is not appropriate for less sophisticated sensors. This means it is necessary to embed a microcontroller within the sensor node to do the sampling, compensation, and communication. In this sense the sensor node becomes not a smart sensor but a complete microcontroller-based system. As a result, a fieldbus "smart sensor" like this suffers from unnecessary complexity and size and high power consumption, and this is certainly not a good choice for microsystem applications.

There is an obvious need for a sensor bus that is better suited for microsystem applications. Although in recent years some steps in the right direction have been made, there is still a gap in providing for networks for smart sensors within a system with sever size and power limitations.

1.2. Background and Motivation

In the past two decades there have been significant breakthroughs in the area of electronic sensors, due in large part to advances in Micro-Electro-Mechanical Systems (MEMS) fabrication technologies and device architectures. With the commercial success of microsensors and microactuators, sensor-based systems are rapidly evolving into sophisticated microinstruments used for distributed gathering of real time physical information. An example of one such system is the Environmental Monitoring System (EMM) which is under development by Canopus System Inc., in Ann Arbor, MI. The EMM, shown in the block-level schematic of Figure 1.1, has an embedded microcontroller which directs the activity of a network of sensor modules. The EMM is designed for the measurement of a variety of environmental parameters with very low power operation from a battery supply (10 year lifetime) and "deck of cards" size limitations. This versatile system can be employed, for example, in general environmental monitoring, industrial process control, and personal health monitoring, and it represents a state-of-the-art microinstrument based on MEMS sensors [2-5].

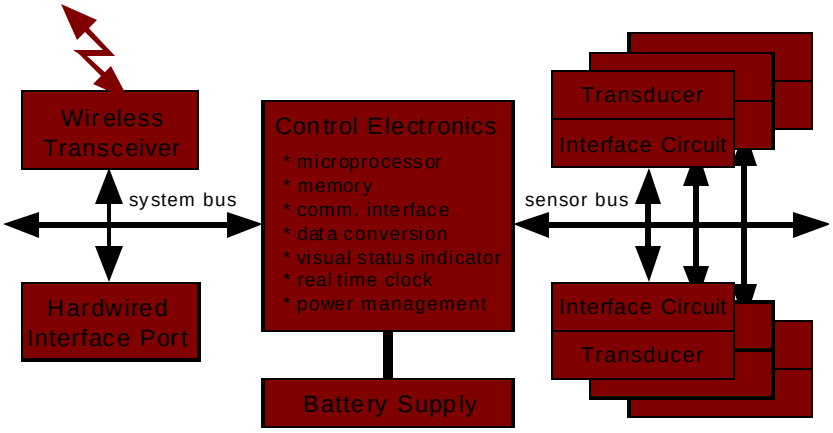


Figure 1.1: Block diagram of the Environmental Monitoring System.

Key elements in this system are the sensor bus which links the control electronics to the multiple sensor modules and the interface circuit which provides the necessary hardware to connect the transducers to the controller. These two elements work together to create a network of high-performance microsensors which can be commanded by and will report to the system controller.

The microsystem architecture shown in Figure 1.1 has been used successfully for microsystems with low power dissipation and very small size [2, 3]. However, the reported system uses a custom sensor bus which is not compatible with any industry standard. It also employs an interface circuit that can only readout capacitive transducers and has limited programmability. Correcting these shortcomings is necessary for the successful implementation of the EMM system and provides the motivation for the effort in this Masters Degree Project to address the unresolved issue of communication and interface to smart sensor modules.

1.3. Research Goals

The objective of this project is to define an appropriate sensor bus and develop an integrated circuit (ASIC) which will provide the necessary interface chip functions for smart sensors employed in microsystems. Design of a sensor bus requires a thorough understanding of both the needs of smart sensors as well as the options and features available in existing communication buses. One goal of this research is to combine the findings in these two areas to create a new Intramodule Multielement Microsystem (IM^2) bus for the network environment of low-power, small size, microsystems. A concurrent objective is to maintain compatibility with an existing standard, if at all possible, in order to provide an improvement on a standard rather than introducing a completely new bus to a market already overwhelmed by bus options. The second major goal of this research is to define an architecture for a Universal Micro-Sensor Interface (UMSI) circuit which can be universally employed in microsystems by a wide variety of transducers. This circuit should be directly compatible with the requirements of small, low-power microsystems. It should also allow such systems to implement advanced features such as programmable sensor readout (gain and offset control), sensor module self test, and temperature compensation, which will support future generations of highly-functionally and highly-adaptable microsystems. In addition to defining the overall architecture for the UMSI chip, an objective of this research is to fully define the bus interface circuitry required to implement the IM^2 bus protocol on the interface chip. This will provide an example of the implementation of the IM^2 bus which helps to highlight is usefulness in microsystem applications. The final objective of this research is to develop a test system capable of generating the IM^2 bus protocol which can be used to interface with the UMSI chips for test and development. Although the full

implementation of the UMSI chip is beyond the scope of the research project, it should be noted that this is being done by another graduate student. This student will generate the physical design of the circuit, characterize the fabricated chip, and modify the design for a final version of the UMSI chip which will be implemented in complete microsystem by the sponsors of this research.

This report begins with the discussion of the requirements for the sensor communication bus and the limitations in existing bus standards. Based on this information, a new sensor bus, called the Intramodule Microsystem Multielement (IM^2) Bus, is presented and defined. The report then details the circuitry necessary to implement the bus on a custom Universal Micro-Sensor Interface (UMSI) chip. The chip architecture, based on the needs of general purpose microsystems, is presented along with details of the design of the interface circuitry which implements the IM^2 bus within the smart sensor module. Finally, a test system which has been designed for test and development of the UMSI chip is presented.

Chapter 2

An Intramodule Multielement Microsystem (IM^2) Bus

2.1. Communication Busses for Sensors

Transducers, defined here as sensors or actuators, serve a wide variety of industry's needs, manufacturing, industrial control, automotive, aerospace, building, and biomedicine are but a few. Since the transducer market is very diverse, transducer manufacturers are seeking ways to build low-cost, networked smart transducers. Many sensor control networks or fieldbus implementations are currently available, each with its own strengths and weaknesses for a specific application class. Interfacing the smart transducers to all of these control networks and supporting the wide variety of protocols require very significant efforts and are costly to transducer manufacturers. However, using digital communication schemes, networked transducers can eliminate much of the lengthy parallel analog wiring and thus reduce the installation, maintenance and upgrade costs of measurement and control systems.

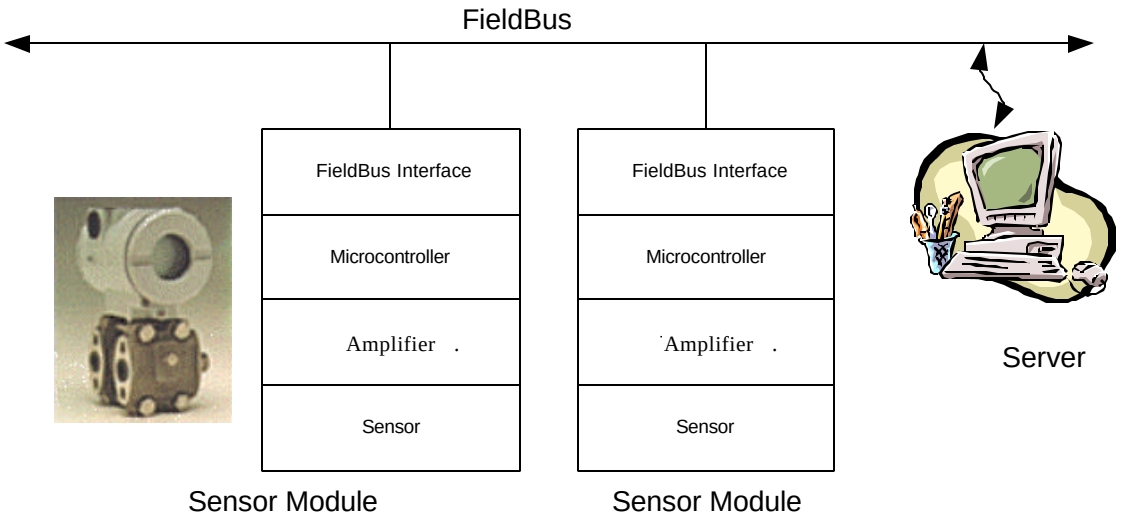


Figure 2.1: Fieldbus Sensor Module Structure and Usage

Many products based on the fieldbus have the ability of being added and exchanged easily. A Fieldbus Sensor Module, see Figure 2.1, is a single monolithic system, which includes a sensor, an amplifier, a microcontroller, and a fieldbus interface. The producer should understand the

whole process from sensor to communication and from hardware to software. The result is that a seemingly simple change from sensor A to sensor B would probably require a wholesale redesign of both hardware and software of sensor module. One would need to discard the sensor, the amplifier, the microcontroller and the fieldbus interface, all at the same time, if one wanted to change the sensor. This would be very expensive [7-9].

A Sensor Bus is used to connect sensors with microprocessors. A sensor bus makes it possible to design each part of the sensor module separately. Sensor producers need not consider how the sensor will be used; they only need to provide the sensor component and relevant data (which can be saved in the EEPROM). Figure 2.2 shows an example of smart sensor module hierarchy. Here, STIM is an ASIC including the transducer interfaces for voltage, resistive and capacitive sensors. STIMs from different vendors and based on completely different underlying technology can deliver exactly the same information to the Microcontroller. Instrument producers will offer the standard product of Microcontroller hardware and software. Finally, users will find it is very easy to build a sensor module that are compatible to any brand product. Users can remove components and plug another band into the sensor module [7-9].

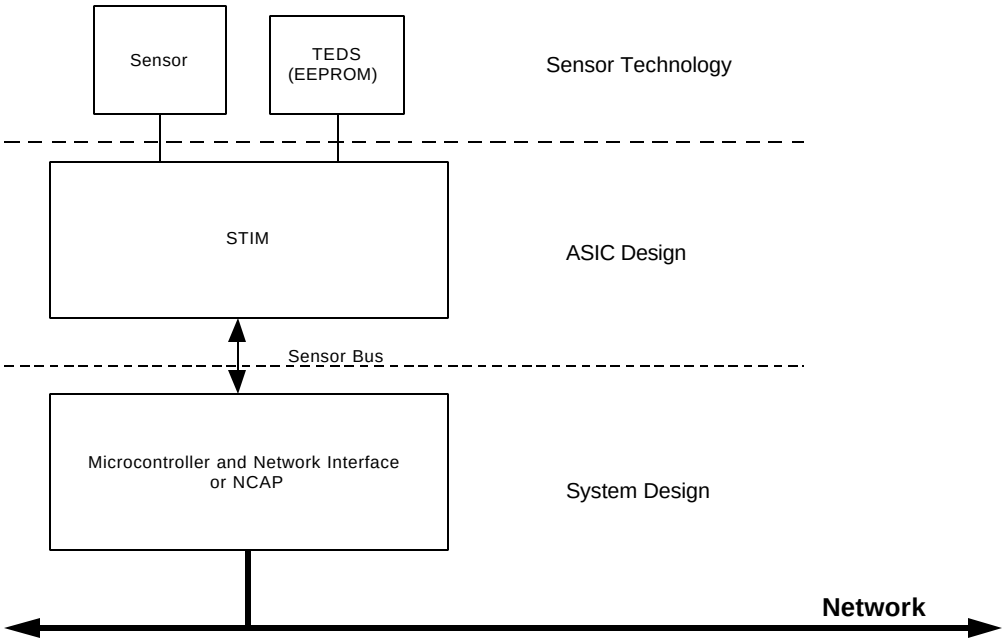


Figure 2.2: Smart Sensor Module Hierarchy

2.2. IEEE P1451 Standard

2.2.1. Description of the IEEE P1451 Standard

The objective of the IEEE P1451 Standard is to develop a smart transducer interface standard which makes it easier for transducer manufacturers to develop smart devices and to interface those devices to networks, systems, and instruments by incorporating existing and emerging sensor and networking technologies. Figure 2.3 illustrates the IEEE P1451 system with the Network-Capable Processor (NCAP), the Smart Transducer Interface Module (STIM), and Transducer Independent Interface (TII).

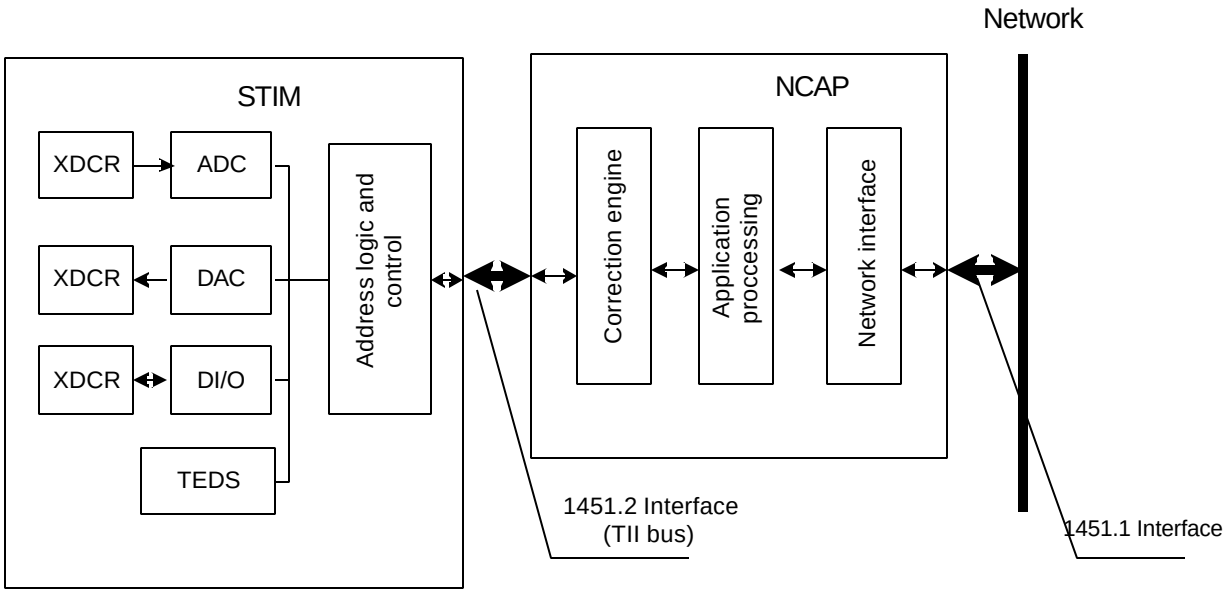


Figure 2.3: Physical representation of IEEE P1451

IEEE P1451.1 Standard

The IEEE P1451.1 Standard defines the Smart Transducer Interface for Sensors and Actuators - Network Capable Processor Information model, which helps to develop a standardized software

framework for connecting the NCAP to control networks. It is similar to many fieldbus products in the market and defines the external interface for an IEEEP1451-based microsystem.

IEEE P1451.2 Standard

The purpose of IEEE P1451.2 [1] is to develop a standard hardware interface to connect transducers to network-capable application processors or microprocessors. The IEEE P1451.2 smart-sensor standard also establishes standard transducer network protocols. In the simplest terms, IEEE P1451.2 specifies a Transducer Electronic Data Sheet (TEDS), which resides within a sensor module, and also specifies a digital interface to access the data sheet, read sensor data, and set actuators. The digital interface is usable by all types of sensors and actuators.

IEEE P1451.3 and IEEE P1451.4 Standard

IEEE P1451.3 and IEEE P1451.4 are draft standards that still wait approval. They do not directly apply to this project and thus no details will be presented here.

2.2.2. The IEEE P1451.2 Smart Sensor Bus

The main features of the IEEE P1451.2 standard are to:

- Enable plug and play at the transducer level by providing a common communication interface for transducers and a machine-readable TEDS.
- Enable and simplify the creation of networked smart transducers.
- The control and data associated with the channel are digital.
- Triggering, status, and control are provided to support the proper functioning of the channel.

Figure 2.3 illustrates a typical system using the IEEE P1451.2 standard. The IEEE P1451.2 transducer, called a STIM, consists of the TEDS, the control and status registers, interrupt masks, address and function-decoding logic, and trigger and trigger-acknowledge functions for the digital interface to the TII. The STIM works in conjunction with a microprocessor-based interface to a communication network. This module, termed NCAP in the standard, implements

network access and the correction of raw data from the STIM. It may also include application specific data processing and control functionality.

Operations that occur in the TII sensor bus are: TEDS read, in which the microcontroller in the NCAP reads the data in the STIM electronic data sheet; sensor read, in which the NCAP accepts data from the sensor; TEDS write, in which the microcontroller writes data (for example, error-correction coefficients) to the TEDS; and actuator write, in which the NCAP sends data to an actuator. In the sensor-read operation, the NCAP triggers the STIM to begin sensor reading. Upon completion, the STIM issues a trigger-acknowledge signal, and the NCAP reads the sensor data. In the actuator-write operation, the NCAP writes values and triggers the STIM. The STIM then performs actuation and issues a trigger-acknowledge signal. The TII bus is the heart and soul of IEEE P1451 which makes it possible for sensors to be "smart" [1].

2.2.3. IEEE P1451 Products Available in Market

Until recently, IEEE P1451 has largely been confined to sensor-industry gurus and insiders. Short of ordering the specifications from the IEEE, there has been little the average embedded-system designer could do to check it out, much less get a head start on development. However, there are now several products available in market which illustrate the use of this standard.

CogniSense EDI520 [6]

The CogniSense EDI520, shown in Figure 2.4, is the product of Electronics Development Corporation (EDC). It is a multichip module consisting of a microcontroller and a mixed-signal application specific integrated circuit (ASIC).

The signal conditioning IC is comprised of an instrumentation amplifier, a reference adjustment digital-to-analog converter, a programmable gain and frequency response filter, an anti-aliasing filter, a temperature sensor, a 10-bit analog-to-digital converter (ADC), and a digital interface. The ADC can be multiplexed to the transducer or the temperature sensor. All of the features of the ASIC can be dynamically reconfigured at any time by the microcontroller.

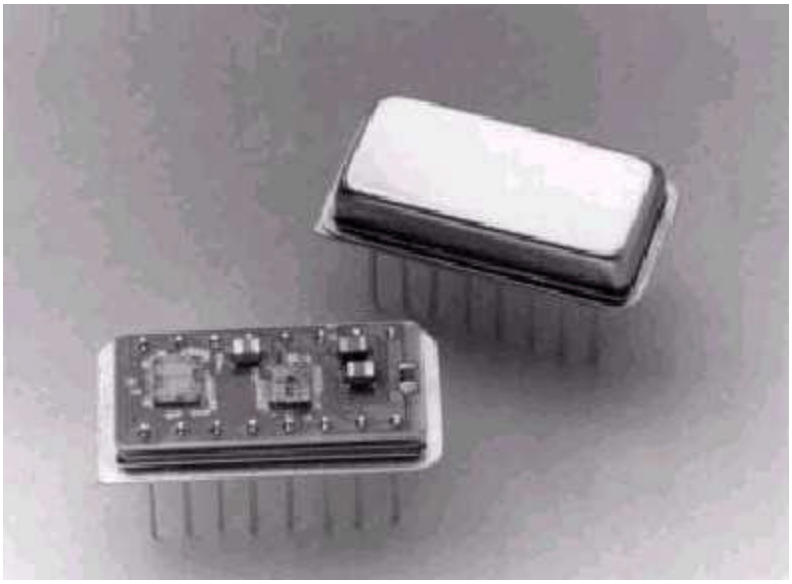


Figure 2.4: The CogniSense™ module combining a PIC and signal-conditioning ASIC can be programmed to function as an IEEE P1451.2 STIM.

The microcontroller in this system is a Microchip PIC16C64 with the I²C/SPI bus and the in-circuit programming pins connected to the EDI520 digital I/O. This allows the EDI520 to use the microcontroller to perform signal processing on the transducer signal, such as digital filtering, threshold sensing, and data integration.

The primary function of the microcontroller and ASIC is the compensation of the transducer for device and temperature induced errors. The EDI520 then allows very precise calibration of the transducer, signal processing capability, decision-making capability and serial interface capability. The level of calibration, processing and decision-making can be determined by the microcontroller software. The serial interfaces readily available with the microcontroller are the I^2C and SPI. Asynchronous communications can be implemented in software. The eight digital I/O pins can be software programmed for any user desired function and can interface with I²C and SPI based EEPROM memory devices for nonvolatile storage of the ASIC settings and other data sheet information.

The PICmicro® MCU as an IEEE P1451.2 Compatible Smart Transducer Interface Module [10]

In an application note from Microchip, a detailed design process has been given for the system shown in Figure 2.5. The PICmicro® MCU is used to design an IEEE P1451.2 Compatible Smart Transducer Interface Module (STIM). Very similar to CogniSense EDI520, the Microchip PIC16C62A and 93C86 EEPROM are the hardware base for the STIM. Other PICmicro® devices could have been implemented for the STIM, such as the PIC16C773 with a 93C86. Another alternative might be to utilize a microchip SPI ™ protocol based EEPROM device in place of the Microwire ® memory device.

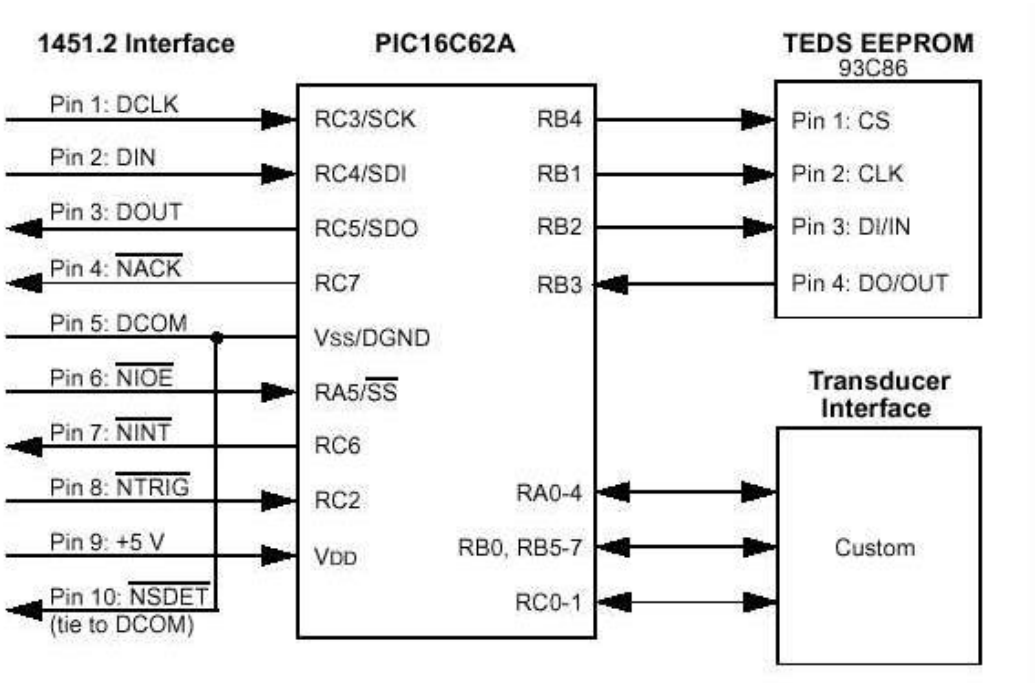


Figure 2.5: STIM Interface Block Diagram with PICmicro® MCU

The requirements should be considered when selecting a PICmicro device for the STIM. For example, the P1451.2 interface requires 8 pins (plus power and ground), the TEDS interface may require up to 4 pins and the transducer interface will most likely require all remaining pins and possibly more. The P1451.2 interface requires a synchronous data transfer with the respective control signals. Selecting a PICmicro MCU with a SPI peripheral will meet this interface requirement. The transducer interface will vary, depending upon the STIM functional requirements. For example, if the STIM requires an ADC, the considerations are as follows:

- 8- 10- or 12-bit ADC

- the required sample rate
- whether the ADC on a PICmicro MCU can be utilized or not
- whether an external stand-alone ADC like the Microchip MCP3201 required or not

Finally, STIM source code has been given in C language, which provides full support for STIM status, interrupt mask, auxiliary status and auxiliary interrupt mask registers.

The ADuC812 as an IEEE P1451.2 STIM [11]

An application note from Analog Device demonstrates how to create a true minimal IEEE P1451.2 standard implementation with Analog Devices ADuC812 MicroConverter. The ADuC812 contains an 8051 compatible microprocessor, 8kb of program flash/EEPROM, 640 bytes of data flash/EEPROM, 256 bytes of RAM, up to 32 programmable I/O lines, a SPI serial I/O port, dual DACs and an 8-channel true 12-bit ADC. The SPI port is an industry standard four-wire synchronous serial communications interface. It can be configured for master or slave operation, and is externally clocked when in slave mode. The data flash/EE is a memory array that consists of 640 bytes and is configured into 160 four-byte pages. The interface to this memory space is via a group of registers that is mapped in the SFR space. The 8kb program flash/EE will ultimately store and run the end users' application code.

The STIM is controlled from the program flash/EEPROM. Each channel's transducer data, status, and control registers are held in RAM for the duration of the STIM lifetime. The transducer interface is mapped onto the ADCs, DACs and I/O lines. The TII is a super-set of the SPI port (plus some I/O lines). The TEDS map into the 640 bytes of data flash/EEPROM, and finally the Address and Function blocks are stored into the program flash/EEPROM.

2.2.4. Problems still exist

IEEE P1451.2 allows sensors to move into the digital age. However, the standard should be simple, inexpensive and easy to use. Refer back the products available in the market which were presented above. Notice the STIM is a complete microcontroller system. The designer of the STIM must understand how to design a microcontroller system and how to design the transducer at same time. This is the same problem facing the existing fieldbus systems. Thus, this standard does not reduce the work of system designer. In this type of system, we might as well do away

with the sensor bus because the microcontroller in STIM can also provide the functions of the NCAP and handle network communication, just like in existing fieldbus sensor modules. In most situations, two microcontrollers will be much more expensive than one, and obviously this configuration will consume more power and remain complex for the system designer.

Even in the face of a hot sensor market, most manufacturers do not design using the IEEE P1451.2. The reason they do not is because it is nearly impossible to implement the standard without an embedded microcontroller which adds significant cost and development time to the sensor product. IEEE P1451.2 requires that the STIM has a repository (E/EEPROM) for the TEDS. This datasheet permits the host to find out what is on the other end of the wire—a sensor system version of plug-and-play. The TEDS is comprised eight fields— two mandatory and six machine readable, each consisting of a length (byte count), data and checksum. Although IEEE P1451.2 defines these in detail to support a wide range of applications, for many products the TEDS is simply too lengthy and complex requiring the added expense of an embedded microcontroller with a significant amount of memory.

As a standard, IEEE P1451.2 is very good. However, every designer is limited by the difficulties stated above. In order to solve these problems, this report will define a new sensor bus and an independent passive sensor bus interface. The developments of this research will remove the reliance on the microcontroller. According to the new sensor bus, the STIM can be a passive ASIC where only the sensor bus clock is needed (no internal clock). This can save a lot of power when the sensor system is working in sleep mode and the sensor bus clock is disabled. In addition, the new sensor bus will support distributed multidrop sensor modules, a feature not currently available with the IEEE P1451.2 standard.

2.3. The Intramodule Multielement Microsystem Sensor Bus

2.3.1. IM^2 Sensor Bus Requirements

The Intramodule Multielement Microsystem (IM^2) Sensor Bus is an expansion of IEEE P1451.2 TII sensor bus which has been developed to correct the shortcomings of the standard. The IM^2 Sensor Bus is an attempt to define a set of specifications that will allow a transducer manufacturer to build transducers that have a wide range of price and performance and can all be utilized within the same system. The IM^2 Bus is designed to have one bus controller (NCAP) and many STIMs to support distributed multidrop sensor module nodes. This bus is also required to support sensor data in both analog and digital formats and offer power management features such as support of normally-off operation. Finally, IM^2 Bus is designed to be superset, or extension, of the TII bus, not a separate standard.

2.3.2. IM^2 Sensor Bus Signals

To achieve the desired features of a multi-sensor microsystem, it is very important to implement a standard interface between the control electronics and the sensor modules that are capable of handling multiple front-end sensor nodes. After careful review of existing standards, the IEEE P1451.2 standard has been adopted. We have expanded this standard to allow for multi-node communication, but the bus is still hardware compatible with the IEEE P1451.2 standard. The signals for this bus are shown in Table 2.1.

As shown in Table 2.1, the electrical connections (Sensor Bus signals) are relatively straightforward. It consists of a clocked serial interface with most of the action revolving around a data clock (DCLK) and unidirectional data lines (DIN and DOUT).

NIOE is an output enable driven by the NCAP that frames activity on the data lines. NACK is driven by the STIM to indicate that a byte transfer can proceed, i.e., that the host can continue to drive DCLK.

Table 2.1: The IM^2 bus and the signals

Line	Logic	Description	Driver
DIN	Positive	Address and data from transport from NCAP to STIMS	NCAP
DOUT	Positive	Data transport from STIM to NCAP	STIM
DCLK*	Positive	Positive-going edge latches data on DIN and DOUT	NCAP

NIOE	Active Low	Signals that data transport is active and delimits data transport framing	NCAP
NTRIG	Negative	Performs triggering function	NCAP
NACK	Negative	Serves two functions: trigger acknowledge and data transport acknowledge	STIM
NINT	Negative	Used by the STIM to request service from the NCAP	STIM
NSDET	Active Low	Used by the NCAP to detect the presence of a STIM	STIM
POWER	N/A	Normal 3-V power supply	NCAP
Controllable Power	N/A	Controllable 3-V power supply. It will be closed when system in sleep mode	
COMMOM/GND	N/A	Signal common or ground	NCAP

*DCLK need not have a constant frequency or duty cycles

Using NACK as a byte handshake reduces the timing burden on the STIM. The timing specification, which is shown in detail below, calls for all devices involved to support a typical 6-kHz DCLK, but it is allowed the components to mutually agree on a higher rate.

NTRIG can be asserted by the host to initiate a particular operation in the STIM. This signal allows sensor establish precise timing when necessary, independent of any latency or uncertainty associated with communication and setup.

For the most part, all activities are carried out under direction of the host, except that NINT can be driven by the STIM to request service asynchronously. However, even in this case, the host response timing is not restricted. Thus, the host is completely in charge, which means that practically any device, fast or slow, can fill the role. NINT can be disabled or the interrupt resource can be reset by a command from the host.

NSDET is also driven by the STIMs as a way to signal STIMs' ID loaded and the new STIM presence. It might be pulled up on the host when no new node adds into the IM^2 Bus. NSDET will be pulled down when all or any node is powered up or re-powered up. Most time, the host may work in sleep mode and close the power of STIMs. When STIMs are powered up again,

each STIM will pull down the NSDET until the host sends the command to ask them to load ID. Load ID has two cases: load ID form IO port and load ID from EEPROM(TEDS). At the case of load ID from IO, ID is wired bond to IO port by user. The host scans the whole possible Ids to find which is occupied. At the case of load ID from EEPROM, EEPROM of new STIM will be blank. When a new STIM is added to the bus, the NSDET will not be restored to pulling up situation until the host allocates a new ID. The new ID will be saved to EEPROM. NSDET is used as a plug-n-play indicator and help the host to manager and allocate the ID.

Although hardware consistency with the IEEE P1451.2 standard has been maintained (i.e., all of the same signals are used with the same logic and function), the data protocol has been modified to fit the needs of microsystems with multiple sensor nodes, while IEEE P1451.2 protocol assumes a point-to-point connection. The primary modifications include:

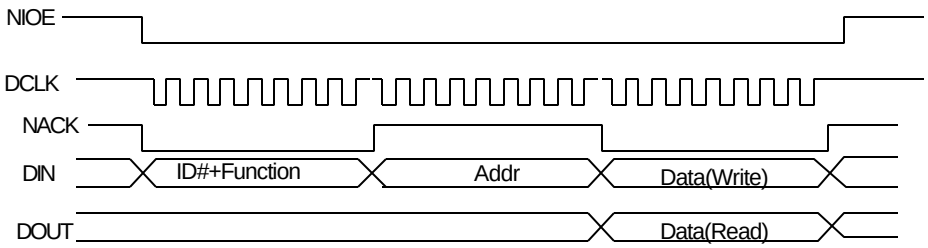
- addition of a chip address byte (ID#) at the beginning of the instruction
- implementation of several modes:
  - compact address mode (chip address and function in one 8-bit word)
  - full address mode (chip address and function in individual words)
  - broadcast mode (for single-chip mode or broadcast to all nodes at the same time)
- modification of the NSDET signal function to serve as a plug-n-play indicator
- definition of DOUT as either digital, analog, or frequency-coded data
- addition of a the second power line to support power management features

Figure 2.6 shows the timing diagram for the data communication signals of the IM^2 sensor bus. Both compact and full address modes are shown. The former can access up to 16 sensor modules while the latter can access up to 255 sensor modules.

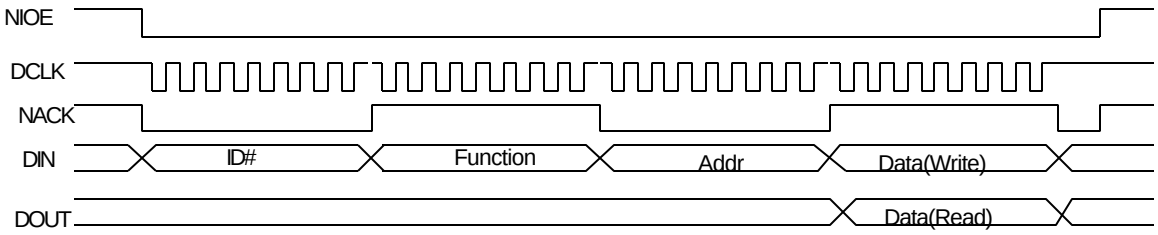
2.3.3. The IM^2 TEDS

In addition to providing an interface to the NACP through the sensor bus, the STIM is required to interface to a memory device in order to implement plug-n-play features. This memory will contain sensor-specific data (TEDS) that can be uploaded by the control electronics. Information such as device type, sensor range/resolution, and calibration parameters can be stored in this

external memory, which allows the control electronics to recognize what devices have been added as necessary for plug-n-play operation. To provide this interface, the STIM implements a Serial Peripheral Interface (SPI) bus, which is a general purpose standard for synchronous serial communications. This provides the STIM not only with the ability to interface to external memory, but also to interface to many generic peripheral devices, such as A/D converters, which can add functionally to the STIM and support of applications that demand such components.



Compact Addr Mode --support 16 ID number



Full Addr Mode --support 255 ID number

Figure 2.6: Timing diagram for the IM^2 sensor bus communication signals.

Chapter 3
Bus Interface Design

3.1. Universal Micro-Sensor Interface Chip Architecture

As previously discussed, the Universal Micro-Sensor Interface (UMSI) is an important component of Environmental Monitoring Microsystem (EMM) [2], and it is also provides an important example for implementing the IM^2 bus protocol introduced by this research. The architecture of the overall UMSI chip is shown in Figure 3.1. The chip includes an IM^2 bus interface unit, a multipurpose analog sensor readout circuit, and an on-chip integrated circuit temperature sensor to provide a highly-programmable interface between the sensor bus and a generic variety of transducer devices. The overall chip includes the following features:

- digital communication with a microcontroller via the IM^2 bus
- readout of capacitive, resistive, and voltage-output transducers
- programmable gain and offset
- amplifier self test features
- on-chip temperature measurement for sensor compensation
- serial interface to external components via an SPI bus

Each of the UMSI chip blocks is a custom CMOS circuit designed specifically to obtain the features listed above. The research project includes the top-level design of the bus interface block in order to demonstrate how the IM^2 bus protocol can be implemented on an interface IC. This chapter will cover the design of the bus interface block.

3.2. Bus Interface Requirements

The UMSI will be the first ASIC with IM^2 Bus interface that really supports smart sensors. The IM^2 Bus Interface Circuit is a critical part of UMSI chip since all information/instructions must pass through this sub-circuit. The bus interface circuit performs the following functions [2-5] which are detailed in paragraphs below:

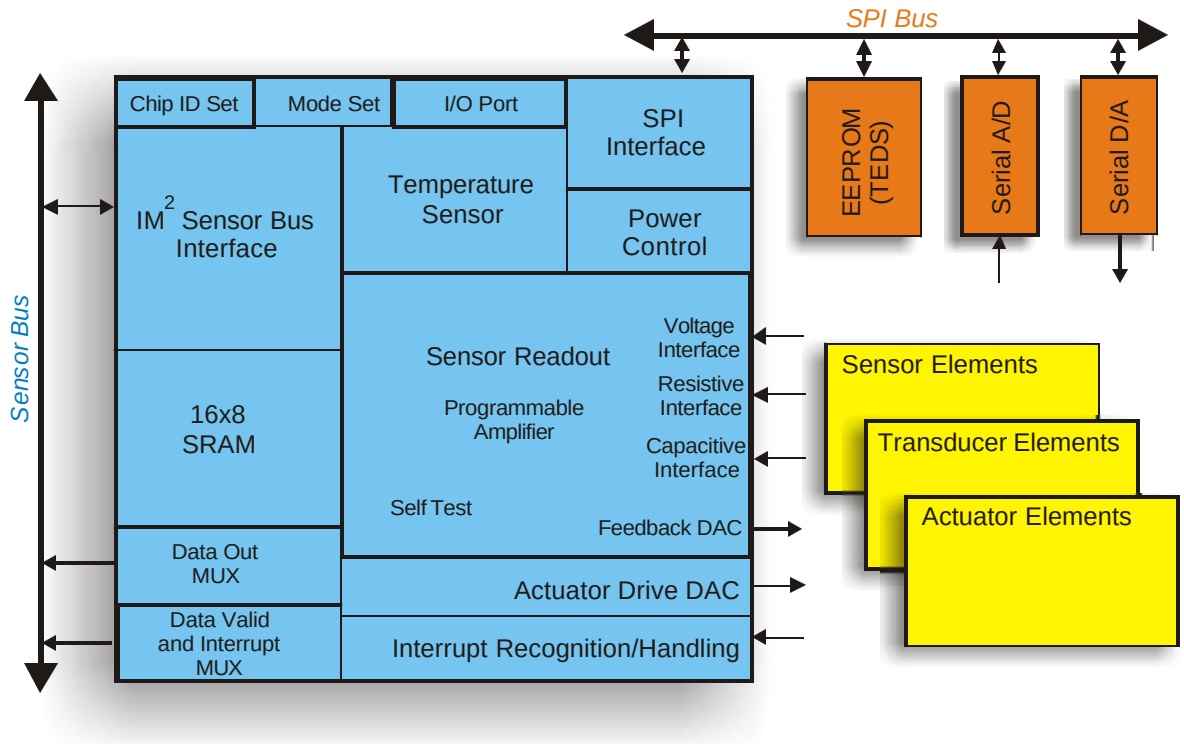


Figure 3.1: Block diagram of the Universal Micro-Sensor Interface (UMSI) Chip.

- interface to the sensor bus and bus protocol handling
- decoding of incoming instructions and transfer of data to other circuit blocks
- read and write access to on-chip memory used to store control data for the UMSI chip
- routing of data through the USMI chip to devices on the SPI bus
- management of the shared data output signal and routing of appropriate output to the bus
- management of data valid and data interrupt signals

IM^2 Bus Compatibility

The UMSI chip provides an interface to the IM^2 sensor bus which supports the design of sensor modules which can be employed in microsystems with suite of front-end sensor nodes. Compatibility with the IM^2 bus allows the UMSI chip to be configured by the host with configuration data stored in on-chip RAM. IM^2 bus compatibility also allows the UMSI chip to support plug-and-play features provided by the bus architecture. The UMSI chip supports the Transducer Electronic Data Sheet (TEDS) feature of the bus via an external EEPROM which is

accessed through the SPI interface on the chip. When a new sensor module is added to the system, the system first reads the TEDS data. This information can be used to configure the UMSI chip for online sensor operation control such as sensor range selection, sensor readout control (gain and offset) and parameter threshold selection (shock sensor). Once configured via commands through the IM^2 bus, the system can then retrieve data from the sensor and calibrate the reading based on calibration and compensation data provided in the TEDS.

Low power Dissipation

UMSI chip is designed to support microsystems which operate under sever power limitations. In such systems, it is assumed that the microcontroller will normally be in low-power sleep mode and will wake to interact with the sensor front end for the following two actions:

- To perform a periodic scan of the sensors, the system will wake up to sample data after a long time sleeping. Once the system finishes this task, a new sleep cycle will begin.
- To provide for sensor-driven interrupts which will wake the system to request attention, UMSI chip utilizes the interrupt line of the IM^2 bus. In this event, the controller wakes up to check which sensor node generated the interrupt signal and then processes the interrupt.

To support these to actions, the IM^2 bus offers two power lines. One is the normal constant, "always on" power line and another is the controllable reference power line. The reference power offers the most power for the UMSI chip and sensor(s), however it can be shut off by the control electronics when the system is in a low power mode. The constant power may be used by circuitry which monitors passive sensors (e.g., switching devices) and will remain powered even when the system is in a low power mode. Further power management of the UMSI chip is provided by having many of the circuit blocks on separate power lines which can be selectively connected (via wire bonds) to enable only the portions of the chip which are needed for a given application.

Application Adaptability

The UMSI chip supports a variety of transducer types via an analog interface which provides readout of capacitive, resistive, and voltage output devices. This allows the UMSI chip to be

employed in a variety of applications where the transducers can be selected to meet the specific needs of each application. Furthermore, the UMSI chip support plug-n-play which allows sensor modules to be added, removed, or updated from a microsystem without affecting the integrity of the overall system. Primary features of the UMSI which support plug-n-play and application adaptability are the use of a standard bus for interface to system control electronics, support of expansion via the external SPI interface, and the support of several working modes such as IEEE P1451 standard interface mode vs. Mixed-signal interface mode and Single Module mode vs. Multi-Module mode. These modes provide not only adaptability to current applications, but also options for different usages in future systems.

Design for Future Expansion

The UMSI design contains a considerable amount, if not all, of the interface electronics needed for a microsystem application on a single chip. This chip can be marketed as a stand-alone microsensor interface solution or as a component in a complete sensor module. Furthermore, the chip provides the flexibility to expand into unforeseen applications by providing a programmable link to external devices such as high-accuracy A/D and D/A components and/or large volume EEPROMS, all of which can be accessed through the UMSI chip via the sensor bus.

3.3. IM^2 Bus Protocol

Protocols describe the implementation of data transport using the physical IM^2 Bus implementation. Both the microcontroller and the UMSI participate in each communication and their individual roles are identified by the protocol. The protocols are hierarchically defined.

The top-level protocols are read frame, write frame, and triggering. These protocols are designed to match with the IEEE P1451.2 standard [1]. The sequence of reading and writing between the controller and the UMSI is illustrated in Figure 2.6. The initial state for all read frame, write frame and triggering protocols had the NTRIG, NACK and NIOE lines negative.

The protocol does not allow the use of DIN and DOUT simultaneously; communication is therefore half-duplex. That is, when data is transferred from Controller to UMSI, the DOUT line

is ignored by the Controller. Likewise, when data is transferred from the UMSI to Controller, the UMSI ignores the DIN line.

3.3.1. Bit Write transfer (Controller to UMSI)

Data shall be transferred in serial bit form from the controller to the UMSI via DIN. The transfer shall be controlled by the DCLK line in the following manner:

a) DCLK idles high
b) After the last rising edge of DCLK, the bit to be transferred is asserted by the sender (the Controller on DIN).
c) On this falling edge of DCLK, the bit is latched by the receiver (the UMSI on DIN).
d) Subsequent bits are transferred by repetitions of steps b) and c)

3.3.2. Bit Read transfer (UMSI to Controller)

Data shall be transferred in serial bit form from the UMSI to the controller via DOUT (for digital signal mode). The transfer shall be controlled by the DCLK line in the following manner:

a) DCLK idles high
b) After the falling edge of DCLK, the bit to be transferred is asserted by the sender (the UMSI on DOUT).
c) Before the next falling edge of DCLK, the bit should be latched by the receiver(the Controller on DOUT).
d) Subsequent bits are transferred by repetitions of steps b) and c)

3.3.3. Byte Write transfer (Controller to UMSI)

All data is transferred from the Controller to UMSI in groups of 8 bits, using the bit write transfer protocol specified above. On the rising edge of last DCLK of 8bits data, the receiver will process the whole byte. The Controller shall proceed with a byte write only after it observes a transition on the NACK line. The UMSI shall cause a transition on the NACK line when it has properly handled the previous byte and is ready for the Controller to proceed.

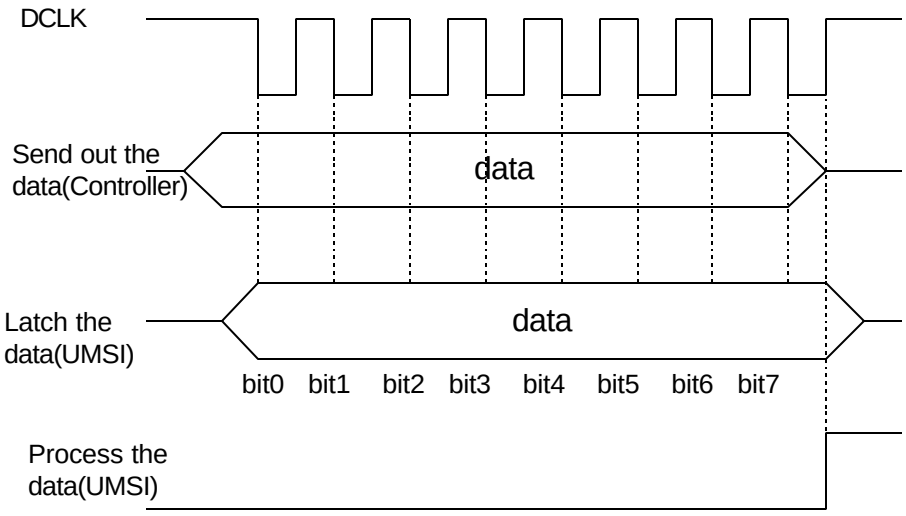


Fig.3.2: Byte Write Transfer

3.3.4. Byte Read transfer (UMSI to Controller)

All data shall be transferred from UMSI to Controller in groups of 8 bits, using the bit read transfer protocol. The Controller shall proceed with a byte read transfer only after it observes a transaction on NACK line. The UMSI shall cause a transition on the NACK line when it has properly handled the previous byte and is ready for the Controller to proceed.

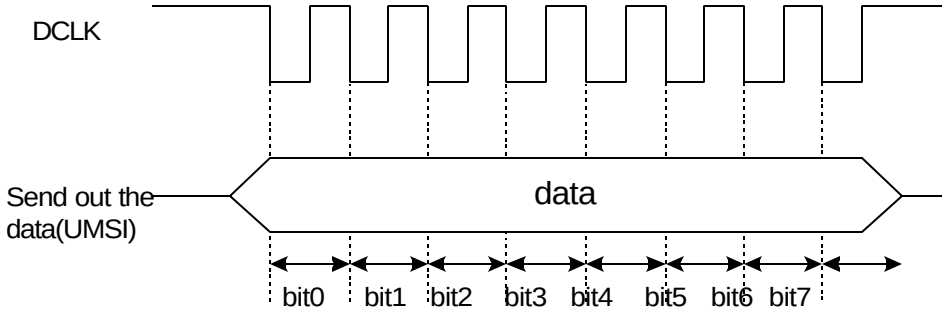


Fig.3.3: Byte Read Transfer

3.3.5. Read frame transfer

The following sequence described the protocol for a Read instruction.

a) Controller asserts NIOE.

b) Controller waits until the UMSI asserts NACK.
c) Controller writes the chip ID using the byte write transfer protocol.
d) Controller writes the function command using the byte write transfer protocol.
e) Controller writes the address using the byte write transfer protocol.
f) Controller reads the data using the byte read transfer protocol.
g) Controller negates NIOE.
h) UMSI negates NACK. (If an odd number of bytes has been transferred, NACK will already be negated due to the byte read transfer protocol).

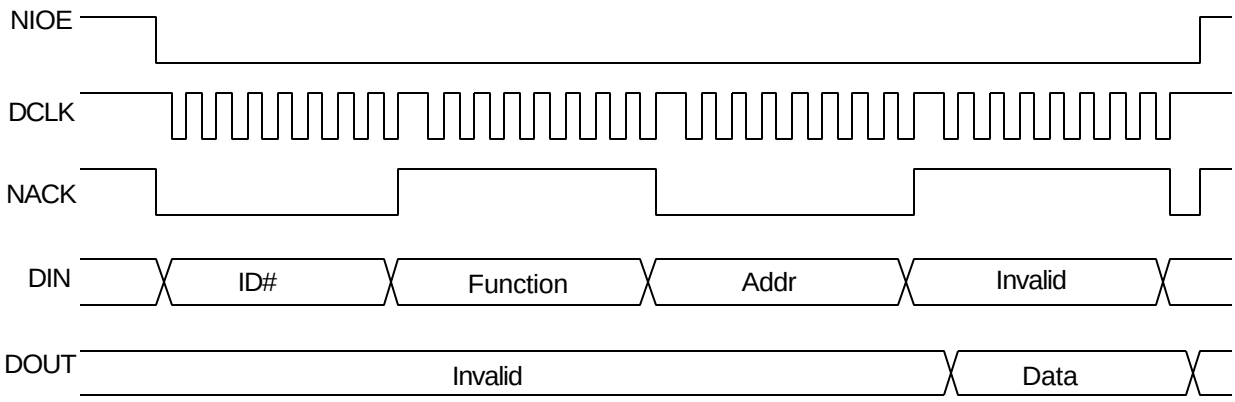


Fig.3.4: Read Frame Transfer

3.3.6. Write frame transfer

The following sequence described the protocol for a Write instruction.

a) Controller asserts NIOE.
b) Controller waits until the UMSI asserts NACK.
c) Controller writes the chip ID using the byte write transfer protocol.
d) Controller writes the function command using the byte write transfer protocol.
e) Controller writes the address using the byte write transfer protocol.
f) Controller writes the data using the byte write transfer protocol.
g) Controller negates NIOE.
h) UMSI negates NACK. (If an odd number of bytes has been transferred, NACK will already be negated due to the byte read transfer protocol).

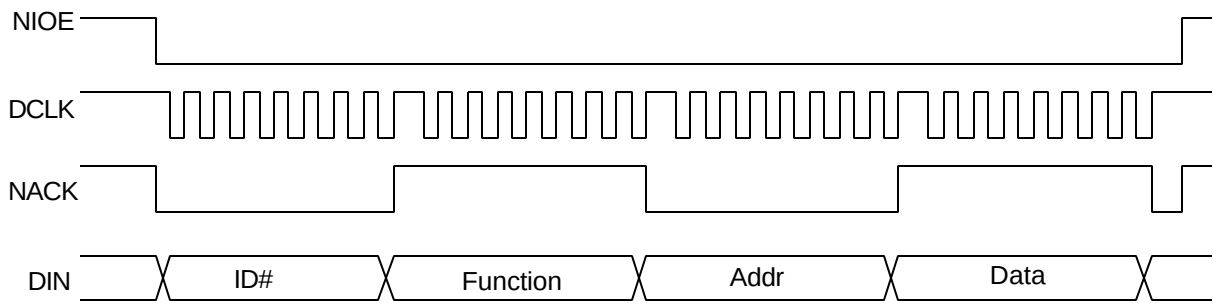


Fig.3.5: Write Frame Transfer

3.3.7. Interrupt Generation and Handling

A separate digital signal in the physical interface is provided to allow the UMSI to request service from the Controller. This interrupt signal is used in conjunction with the UMSI chip Status Register and Interrupt Mask Register to indicate exceptional conditions have occurred. When servicing an interrupt, the Controller should always read the status register which will specify the interrupt sources. The mask register is used to disable some of the interrupt sources. The interrupt can signal the Controller during normal operation to respond to the interrupt. During times when the Controller is operating in a low-power sleep mode, the UMSI interrupt will wake the system when sensor detected conditions warrant immediate attention. Examples of this include a physical shock or other sensor signal over a preset limit being monitored by the sensor module(s). This feature is especially important for very low power systems where it is expected that the system will spend most of its time in sleep mode to save battery energy.

The process of interrupt is:

a) UMSI: NINT-High=>Controller: UMSI sends Controller an interrupt signal via NINT when internal UMSI circuitry or an external source generates an interrupt.
b) Controller: NIOE-High=>UMSI: Controller sends NIOE high to stop all operation.
c) Controller: reads the UMSI interrupt status register to find the source of the interrupt.
d) Controller: response the interrupt according to control software routines

e) Controller: writes the UMSI interrupt mask register to disable the interrupt (in case of an internal source) or writes an I/O port to create a Reset signal for the interrupt (in case of an external source).

Interrupt Masks

The UMSI chip contains an Interrupt Mask register for controlling the threshold sensor interface interrupts (4 bits) and the external interrupt (1 bit). Writing a '1' to any Interrupt Mask bit position in will allow an interrupt signal to be generated on the DINT sensor bus pin when the corresponding bit in the Status Register is set.

Interrupt Status

Interrupt Status shows which sensor generates the interrupt. It can be readout by ReadMem instruction.

3.3.8. Triggering

Signal triggering allows the Controller to send a timed gate signal to the sensor front end via the NTRIG signal lines on the physical bus. This allows actions within the UMSI chip to be timed and synchronized by the Controller. The trigger signal is applied to all nodes on the bus at the same time. A Trigger Enable control bit in UMSI memory determines whether the trigger signal will be passed through that specific chip. The trigger signal is utilized by the UMSI chip to activate and stop the temperature sensor counter thus providing an accurate gate timing to control readout of this sensor.

3.4. Instructions Description

The list of instructions (organized by *function* byte) and their operations are summarized in Table 3.1. All function, address and data bytes are transferred with the MSB first. In order to support two addressing modes, all function bytes use only the lower 4 bits of the function byte.

Table 3.1 Instructions Set for UMSI chip

Instruction Name	Function Byte Code	Operation
ReadMem addr,data	xxxx0000b	Read RAM data from UMSI
WriteMem addr,data	xxxx0001b	Write data to RAM
ReadIO addr	xxxx0010b	Read I/O
WriteIO addr, data	xxxx0011b	Write I/O
LdID_IO	xxxx0100b	load ID from I/O
LdID_ROM	xxxx0101b	load ID from EEPROM
RdSensor	xxxx0110b	Read Internal Sensor in the UMSI chip
OutMixed	xxxx0111b	Send the mixed signal to Dout
TII_SPI	xxxx1000b	Transfer data with SPI bus

Read Data from Memory (ReadMem address, data)

Controller uses Read Frame transfer protocol to send ID+Function+Address message**(see fig. 3.4)** and receives the data returned from UMSI chip. The Address byte determines which memory address on the UMSI chip will be read out.

Write Data to Memory (WriteMem address, data)

Controller uses Write Frame transfer protocol to send ID+Function+Address+Data message**(see fig. 3.5)**. UMSI receives it and saves the data in the given address memory.

Read the status of I/O port (ReadIO)

Controller uses Read Frame transfer protocol to send ID+Function+Address message and receive the I/O data return from UMSI chip. To current UMSI, Only one 8 bits I/O port exists. So address can be any value.

Write Data to I/O port (WriteIO)

Controller uses Write Frame transfer protocol to send ID+Function+Address+Data message. The UMSI receives the message and sets the I/O port according to the data. The appropriate I/O

Direction Control bits must be set **(high for input and low for output**) in order to enable output of the data written to the I/O port. Only one 8 bits I/O port exists. So address can be any value.

Load ID from I/O port (LdID_IO)

Controller uses Byte Write transfer protocol to send ID+Function message, and the UMSI responds by loading a 4-bit ID from the ID I/O pins into the ID Resister. Each UMSI chip should have an independent ID number which serves as its bus node address. Because the UMSI does not have non-volatile memory an external source is required to store the chip ID when the chip is powered down. The UMSI supports two options for setting the chip ID; a hardwired option where the chip ID is set by discretionary wire bonds at the ID I/O pins (this instruction) and an external memory option where the ID is load through the SPI interface (next instruction). When the UMSI is a newly added one in the system (Power Up), the DSDET of this UMSI will be low. After UMSI receives the LDID_IO, the DSDET will be reset to High level.

Load ID from EEPROM (LdID_ROM)

Controller uses Byte Write transfer protocol to send ID+Function message and UMSI loads the ID from an external memory (e.g., EEPROM). When Controller sends this function, the UMSI will change to SPI mode and monitor the communication between Controller and EEPROM to capture the ID data. Additional information regarding the options for loading the chip ID is given under the *LdID_IO* function.

EEPROM of new STIM will be blank. When a new STIM is added to the bus, the NSDET will not be restored to pulling up situation until the host allocates a new ID. The new ID will be saved to EEPROM via writing SPI EEPROM. After that, host will send LDID_ROM again to specific address (0x00) to load ID again. This time, the ID will be updated to the allocated ID.

Send the Mixed Signal to Dout Pin (OutMixed)

Controller uses Byte Write transfer protocol to send ID+Function message and UMSI sets a switch to enable mixed signal output to the DOUT pin. Mixed signal may be the voltage output of on chip amplifier or frequency signal of temperature sensor or one of external analog inputs.

Transfer Data Between IM^2 Bus and SPI bus (TII_SPI)

TII_SPI is used to transform the IM^2 Bus into SPI mode to allow the Controller to communicate directly with any external device connected at the SPI port of the UMSI.

Controller uses Byte Write transfer protocol to send ID+Function message and UMSI sets a switch to TII_SPI mode. The SPI node will be selected at the same time. But the data for selecting which channel should be preset in memory via WriteMem instruction before implementing TII_SPI.

After CPU receives the NACK of AddrEnd, Controller can send and receive SPI protocol data. When the NIOE becomes to High Signal, the UMSI closes the SPI bus and transforms back to IM^2 Bus.

3.5. Addressing

The UMSI chip includes a RAM block and an I/O register which provide memory that allows the UMSI to be programmed by the Controller. The majority of the RAM is used to control the highly-programmable analog sensor readout circuitry of the chip. .Mem[16], Mem[17], Mem[26], and Mem[27] are status-readout bytes

Sensor Bus Interface:

Mem[0]: ***Analog Select and SPI Chip Select Control Register***

1. Bits [7:4] SPI Chip Select. It will be loaded to the I/O register. Each time, only one SPI chip will be selected. Before use the SPI chip, we need to write this memory unit to make sure the correct SPI chip is selected. [7:4] match IO port[3:0]. When Controller sends "TII/SPI" function, the IO port[3:0] will be used as SPI select signal.
2. Bit [3] Reserved.
3. Bits [2:0] are used for selecting one from 8 mixed channels. The selected channel will be send to "AnalogOut" pin. At the same time, it will be sent out from "Dout" pins when Controller sends "OutMixed" function.

Bits[2:0]=000b-111b match the mixed channel 0-7. Some channel share the input pins with other function to input the external digital sensor and sensor with amplifier.

Chn 0: voltage output of on chip amplifier;

Chn 1: frequency signal of temperature sensor;

Chn 2-3: share input pins of sensor inputs [0-1];

Chn 4-5: share input pins of sensor inputs [4-5];

Chn 6: share shock sensor input pin[0];

Chn 7: share shock sensor input pin[4];

Mem[1]: ***Preset of ID Capture Counter Register***

ID capturing from EEPROM is an important issue in UMSI. Several ways are designed to make it flexible. When Controller sends "LdID_ROM" function, Controller will continue to send the SPI command to read data from EEPROM where the ID of UMSI is saved. At the same time, UMSI needs to monitor every bit from EEPROM and find the start and the end of ID information. Preset of ID Capture Counter is used to adjust the beginning time to capture the ID. We design a default constant number for counting. At the same time, the preset number can be set on the RAM. SPI EEPROMs have different length of address. So it is necessary to design the programmable preset for ID Capture Counter. SPI EEPROM has three address modes: normal mode includes 16bits address, small mode includes 8 bits address and large mode includes 24 bits address.

- Bit [7], "1", enables the Preset of ID Capture Counter; "0", selects the constant default.
- Bit [6:0], the preset of ID capture counter. The ID register will begin to capture the ID when the clock number equals to the preset. After 8 clocks, the ID register will stop capturing.

Mem[2]: ***I/O Direction Control Register***

The UMSI chip has a generic 8-bit I/O port, and each bit can be independently configured as input or output. The direction of each pin is determined by the I/O Direction Control Register where '1' enables the output and '0' defines the pin as an input. The direction control register should be set before any instruction to write to the I/O port.

I/O Direction Control Register will be enabled the I/O as output pin only after receiving the command of WriteIO.

The UMSI has an 8-bit I/O port used in 4 ways: wired bond ID selected, SPI chip select, I/O input and I/O output. IO Direction Control is used to set which channel can be read or written. I/O Direction Control [7:0] matches the I/O pins[7:0].

When the Controller sends the "WriteIO data", I/O Direction Control should be set before this function to activate the pins can be used as output. Data [7:0] match to I/O pin[7:0]. "1" means this activated I/O output should be High; "0" means this activated I/O output should be Low. I/O Direction Control bits are "0" that means these I/O only can be used as output I/O and "1" means input I/O.

Mem[16] & Mem[17]: ***Interrupt Status (Read Only)***

The register shows the status of each of the interrupt sources and is a read only register. For each bit, '1' indicates the source associated with that bit has generated an interrupt. The status will be locked into the register when its interrupt line change from "0" to "1".The register will be cleared by the clear bit of Interrupt Masks.

- Bit [7], external interrupt. "1" means that the interrupt signal comes from an external connected at the external interrupt pin of the UMSI.
- Bits [6:0], threshold sensor interrupts.

Mem[3]: ***Interrupt Masks***

This register allows each interrupt source to be masked so that any interrupts generated by that source will be ignored. i.e., they will not cause the NINT pin to be set nor will they affect the Interrupt Status Register.

- Bit [7], external interrupt mask. "1" means enable, "0" means disable.
- Bits [6:0], threshold sensor interrupts. "1" means enable, "0" means disable.

Temperature Sensor:

Mem[4] & Mem[19]: Temperature Sensor Control Register

Mem[26] & Mem[27]: Temperature Sensor Status

Analog Sensor Readout Interface:

Mem[7] Digital-to-Analog Register 1

Bit [6: 0], DAC digital input for reference voltage Varef

Mem[23] Digital-to-Analog Register 2

Bit [6: 0], input signal of DAC for self-test to generate Vp

Mem[8] Amplifier Channel Select Register

Bit [4: 0], 8-to-1 Multiplexer Select. It decides which sensor output will be read out.

Bit [6: 5], 4-to-1 Multiplexer Select. It decides which readout channel will be amplified by second gain stage.

Mem[12]: Reference Capacitor Array Switch Register

This register is used to control reference capacitor array Cref in capacitive readout circuit.

Mem[13]: Attenuator Array Switch Register

Bit [5: 0], It is used to control programmable attenuator b0~b5 in voltage readout circuit.

Mem[14]: Feedback Capacitor Array Switch Register & Resistor R1 Control Register

Bit [6: 4], It is used to control programmable feedback capacitor array in second gain stage.

Bit [3: 0], It is used to control resistor R1 in resistive readout circuit.

Mem[15]: Resistor R2 Control Register

This register is used to control resistor R2 in resistive readout circuit.

Mem[25]: Clock Phase Control Register

Bit [6: 0], It is used to generate 7-bit clock phase with clock generator to control input capacitor array Cin in second gain stage.

3.6. SPI port

The UMSI chip includes a SPI port to interface to external components to extend the functions of the UMSI. Serial EEPROM can be used to store the TEDS including the UMSI chip ID. An external serial A/D can used to sample the analog sensor signals and provide sensor module output in a digital format. A serial D/A can be used to generate analog voltages used as set points for analog circuitry within the UMSI or as input to actuator devices. The SPI interface give the UMSI the flexibility to be used in many applications which are not directly supported by hardware on the UMSI chip.

3.6.1. SPI bus Overview

The SPI is essentially a three-wire serial bus for eight or sixteen bit data transfer applications. The three wires carry information between the devices connected to the bus. Each device on the bus acts simultaneously as a transmitter and a receiver. Two of the three lines transfer data (one line for each direction) and the third is a serial clock. Some devices may be only transmitters while others only receivers. Generally, a device that transmits data usually possesses the capability to receive data as well. An SPI display is an example of a receive-only device while EEPROM is a receiving and transmitting device.

The devices connected to the SPI bus may be classified as master or slave devices. A master device initiates information transfer on the bus and generates the clock and control signals. A slave device is active only when selected by the master via a slave select (chip enable) line which is an additional signal to the standard three-wire SPI bus. Generally, a dedicated select line is required for each slave device. The same device can possess the functionality of a master and a slave, but at any point of time only one master can control the bus in a multi-master mode configuration. Any slave device that is not selected must release (make high impedance) the slave output line.

The SPI bus employs a simple shift register data transfer scheme: data is clocked out of and into the active devices in a first-in, first-out fashion. It is in this manner that SPI devices transmit and receive in full duplex mode.

All lines on the SPI bus are unidirectional: The signal on the clock line (SK) is generated by the master and is primarily used to synchronize data transfer. The master-out, slave-in (SIN) line carries data from the master to the slave, and the master-in, slave-out (SO) line carries data from the slave to the master. Each slave device is selected by the master via individual select line. Information on the SPI bus can be transferred at a rate of near zero bits per second to 1 Mbits per second. Data transfer is usually performed in eight or sixteen bit blocks. All data transfer is synchronized by the serial clock (SK), and one bit of data is transferred for each clock cycle. Four clock modes are defined for the SPI bus by the value of the clock polarity and the clock phase bits. The clock polarity determines the level of the clock idle state and the clock phase determines the clock edge that places new data on the bus. Any hardware device capable of operating in more than one mode will have some methods of selecting the value of these mode control bits. This multi-mode capability combined with the simple shift-register architecture makes the SPI bus very versatile and allows many non-serial devices to be used as SPI slaves. STB is the hold signal to suspend any serial communication without resetting the serial sequence.

3.6.2. Instructions for SPI Operation

The bus instructions which manage SPI operation include, generate SPI chip-select signal, read/write data between Controller and SPI device, and retrieve bus address (chip ID) from EEPROM.

SPI chip-select signal generation

The SPI chip-select signals can be generated using the 4-bit UMSI I/O port which are programmed by the Controller to select the device it wants to communicate with. Before a SPI read/write operation, the I/O Direction Control Register needs to set and then the SPI Chip Select Control Register must be set to send a chip-select output signal for the chosen SPI chip. These commands will prepare the SPI device for direct communication with the Controller over the IM^2 bus.

Read/Write Data between Controller and SPI chip

The Controller sends out the "TII_SPI" instruction which selects the desired sensor module (via the chip ID) and then essentially transforms the IM^2 bus to a SPI bus where DIN will link to SIN,

DOUT to SO, DCLK to SK and NACK to STB. This allows the Controller to communicate with the SPI device directly. This function will be closed at NIOE rising edge. Figure 3.2 shows the timing for an IM^2 SPI Read/Write operation

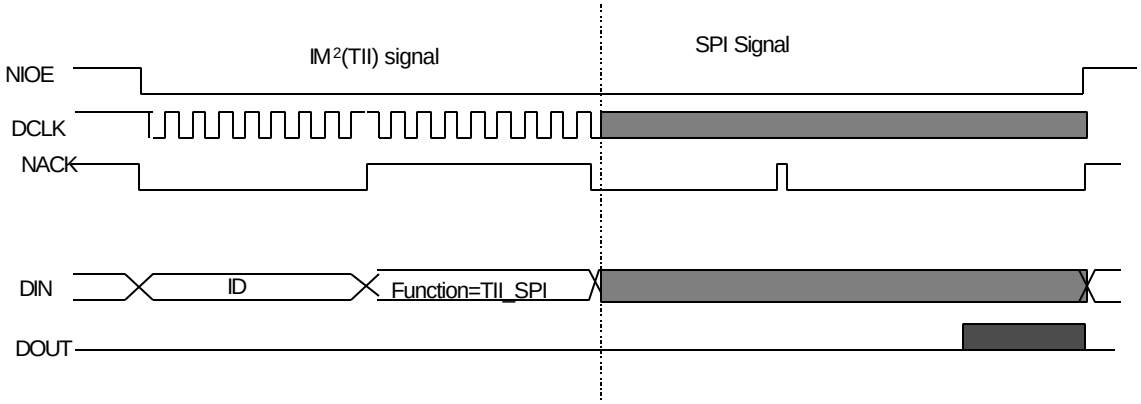


Figure 3.2: IM^2 SPI Read/Write operation

Retrieve Chip ID from EEPROM

Each time an UMSI chip is powered up it needs to load the chip ID from I/O port or EEPROM to identify itself among the different chips on the IM^2 Bus. To retrieve the chip ID from external memory, the Controller uses the "LDID_ROM" command (using the broadcast address: 0xFF) to load the ID from EEPROM. This operation, which is shown in Figure 3.3 is similar to the "TII_SPI" operation; however, no DOUT signal is enabled from the DOUT pin. The "LDID_ROM" instruction, unlike the "TII_SPI" operation, requires the UMSI to monitor the SPI SO signal (from EEPROM) in order to capture the ID data. This requires the UMSI chip to count cycles of SCLK in order to determine exactly when the ID information will be on the SO line. Finally the ID data is saved in an UMSI register and used as the preset chip ID for the UMSI chip. Notice that this command allows the chip ID of all chips on the bus to be retrieved at once, using only one instruction from the Controller.

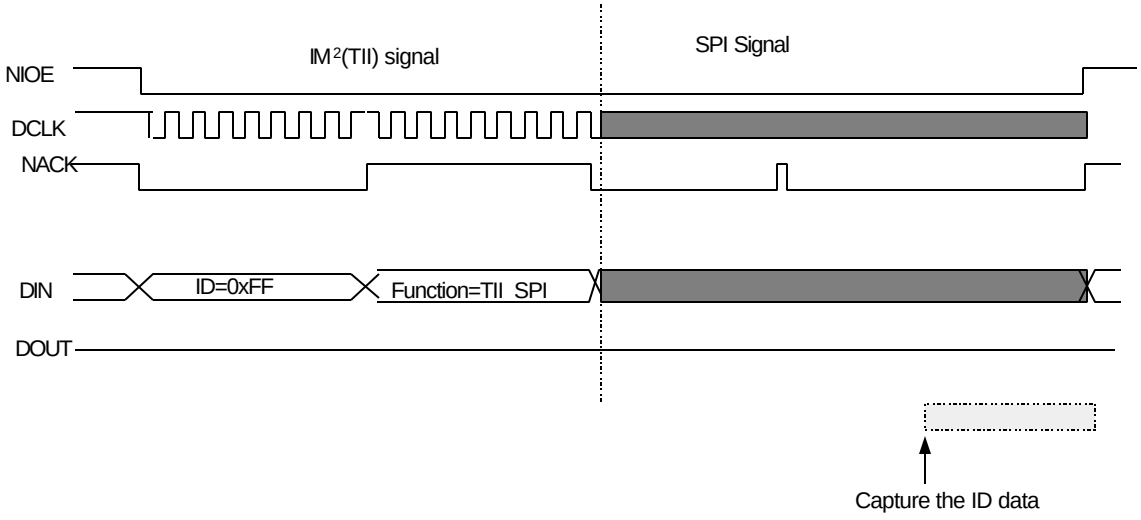


Figure 3.3: UMSI gets the ID from EEPROM

3.7. Plug-n-Play

3.7.1. New Node Detection

NSDET is also driven by the STIMs as a way to signal STIMs' ID loaded and the new STIM presence. It might be pulled up on the host when no new node adds into the IM^2 Bus. NSDET will be pulled down when all or any node is powered up or re-powered up. Most time, the host may work in sleep mode and close the power of STIMs. When STIMs are powered up again, each STIM will pull down the NSDET until the host sends the command to ask them to load ID. Load ID has two cases: load ID form IO port and load ID from EEPROM(TEDS). At the case of load ID from IO, ID is wired bond to IO port by user. The host scans the whole possible Ids to find which is occupied. At the case of load ID from EEPROM, EEPROM of new STIM will be blank. When a new STIM is added to the bus, the NSDET will not be restored to pulling up situation until the host allocates a new ID. The new ID will be saved to EEPROM. NSDET is used as a plug-n-play indicator and help the host to manager and allocate the ID.

When any new node is added into the IM^2 Bus or system powers up, the NSDET will be pulled down to tell Controller that nodes need to load ID. Controller needs to broadcast LDID_IO or LDID_ROM command to load ID saved in IO port or EEPROM. After loading the ID data, most time the NSDET request will be cancelled by the UMSI chip. But, to loading ID from EEPROM,

if the ID in EEPROM is blank(new node), NSDET will be cancelled until Controller allocates a new ID. ID in EEPROM is composed of 1 status bit(ID_status) plus 7 ID bits. If ID_status bit is “0”, that means no valid ID allocated for this node. A valid ID should be from 0 to 127 and ID_status should be “1”.

3.7.2. TEDS

The Transducer Electronic Data Sheet (TEDS) allows a system to automatically load the calibration constants for a particular transducer and sensor. As an electronics datasheet, TEDS permits the host to find out what is on the other end of the wire—a sensor version of plug and play.

The EEPROM will be used to save the TEDS. IEEE standard has detail definition of the data structure of TEDS. The Controller can reads/writes the TEDS (EEPROM) to perform the Plug and Play function.

3.8. Embedded Test Port

Similar to the JTAG port, UMSI includes an embedded testing port for boundary-scan which is shown in Figure 3.4. Four pins of the UMSI chip provide the testing port: tclk, tload, tdin, tdout. tclk is the clock input pin for the test port. tload offers the raising edge signal to latch the signals under test. tdin is the data shift in pin. tout is the data shift out pin.

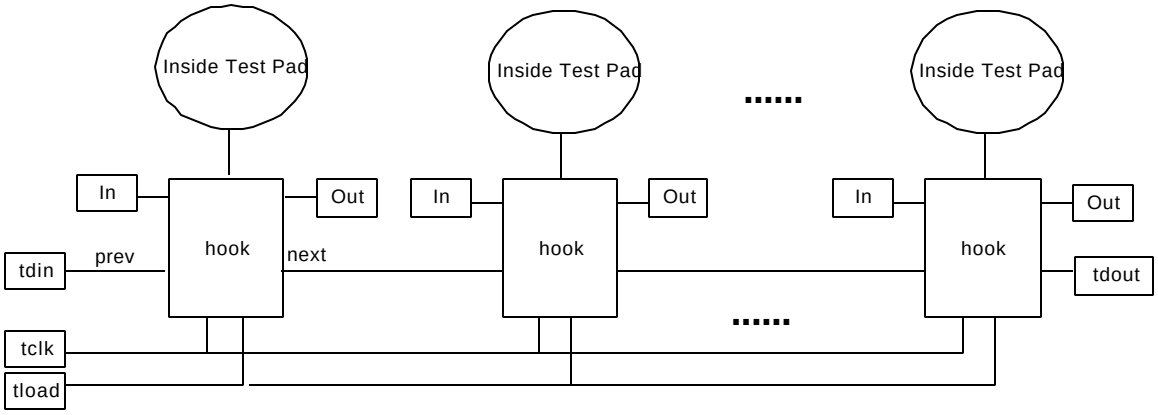


Figure 3.4: Testing chain

When tclk=0, “Out” links directly to “In”. (In ⇔ Out)

When tclk=1 and “next”(or Q)=1, Test Pad links to “Out”. (Tstpad⇔ Out)

When tload: 0->1, “Out” data will be latched and saved in D flip-flop. So if we add tclk, the data can be shifted out; if we want to set data of D flip-flop, we can input data from tdin.

When tload=1, Test pad links to “In”. (In⇔ Tstpad)

Applications:

tclk=0 and tload=0: UMSI works in normal mode

load=1: Test the output of previous module

tload=0, tck=1 and Q=1: Add test signal for next module from test pad. Q is shifted bit by bit from tdin. So we can enable(Q=1) or disable(Q=0) the test nodes that can add test signal.

tload 0->1 and tclk is added: Read out the "In" status. Sometimes, it is also used as the Q shift-in. The data for shift-in is from tdin that can enable or disable the node function of adding test signal for next module.

Figure 3.5 is the schematic of one testing node (hook) on the scan ring.

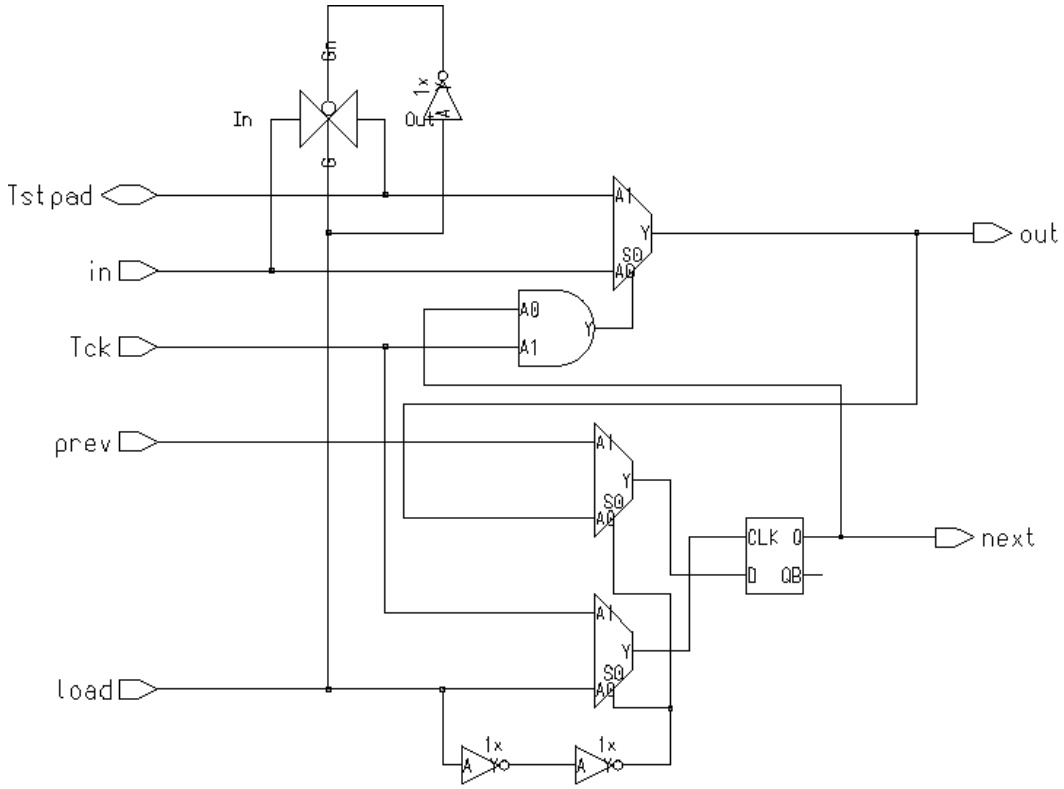


Figure 3.5: The schematic of one testing node (hook)

3.9. The UMSI chip Usage

The UMSI chip can work in three modes that provide users with system flexibility. These are described below.

3.9.1. Single Node without ID

When the UMSI works in this mode, as shown in Figure 3.6, it is similar to the mode defined by IEEE P1451.2. Because the chip address byte is not required in this configuration, compact address mode is used and the controller uses the broadcast ID to directly communicate with the single UMSI chip.

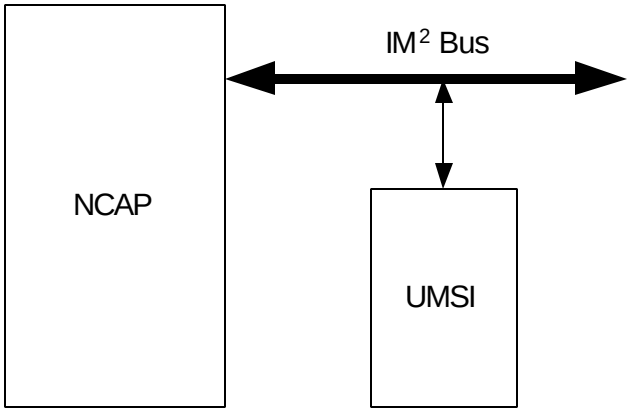


Figure 3.6: Single Node without ID

3.9.2. Multiple Nodes with ID loaded from I/O port

This mode, illustrated in Figure 3.7, supports up to 16 sensor nodes where the chip address is determined by the 4-bit ID I/O port on each UMSI. Since the ID is only 4 bits, compact address mode should generally be used in this configuration.

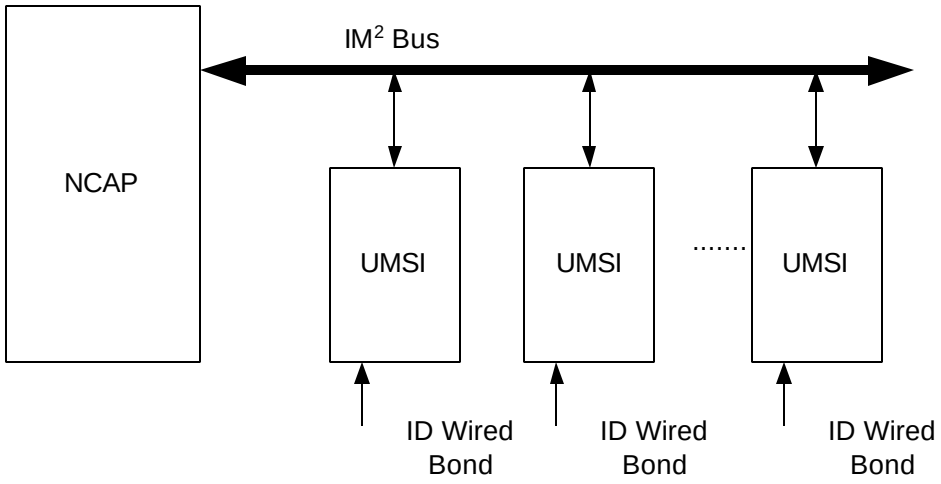


Figure 3.7: Multiple Nodes with ID loaded from IO port

3.9.3. Multiple Nodes with ID loaded from TEDS (EEPROM)

In this configuration, which is shown in Figure 3.8, the chip ID for each UMSI/node is saved in memory (e.g., EEPROM) within each module that provides the TEDS. Here, up to 255 sensor nodes can be used and the UMSI chips are operated in full address mode.

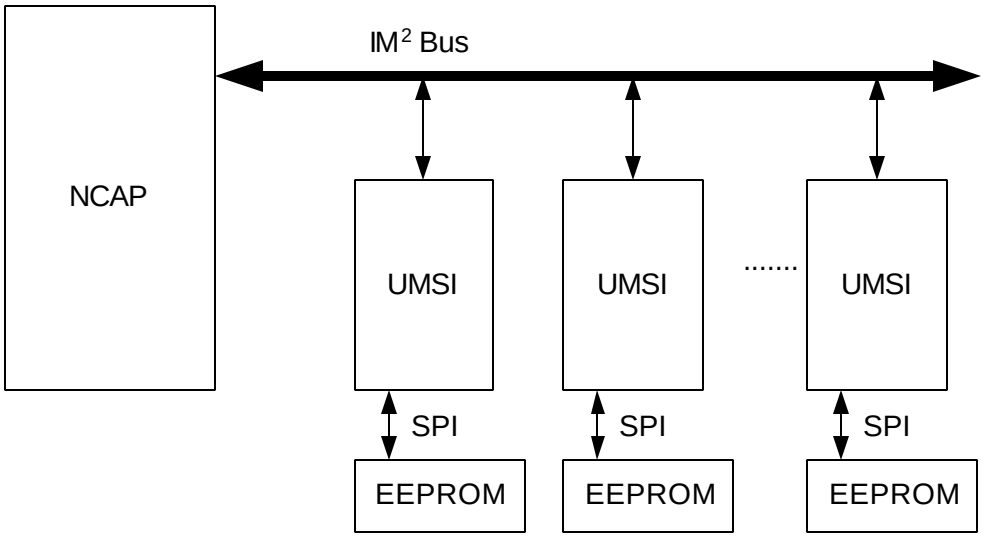


Figure 3.8: Multiple Nodes with ID loaded from TEDS (EEPROM)

Chapter 4
Test System Design

4.1. IC Test System Overview

After the UMSI chip is designed and fabricated it must be fully tested. However, due to the fact that most of the functions of the chip are programmably controlled via the IM^2 bus interface, it is vital to have instrumentation setup to generate and manage this bus.

4.2. Hardware Design

4.2.1 Diagram of IC Test System

Figure 4.1 shows the test system. The Texas Instruments MSP430F149 is a very low power microcontroller which will serve as the Controller according to the IEEE P1451 format. Several I/O pins and internal timers and interrupts of this controller are used to create the IM^2 bus and communicate with the UMSI chips under test. The user can interface with the controller via a PC with an RS232 connection to the MSP430F149 controller. This allows the microcontroller to generate the necessary sensor bus timing while the PC simply interfaces between the user and the microcontroller.

4.2.2. MSP430 series microcontroller works as the Controller

The Texas Instruments MSP430 series is an ultra-low-power microcontroller family with several devices featuring different sets of modules targeted to various applications. The microcontroller is designed to be battery operated for use in extended-time applications. The MSP430 achieves maximum code efficiency with its 16-bit RISC architecture, 16-bit CPU-integrated registers, and a constant generator. The digital-controlled oscillator wakes up CPU from low-power mode to active mode in less than 6μs. The MSP430x14x series are microcontroller configurations with two built-in 16-bit timers, a fast 12-bit A/D converter, two universal serial synchronous / asynchronous communication interfaces (USART), and 48 I/O pins.

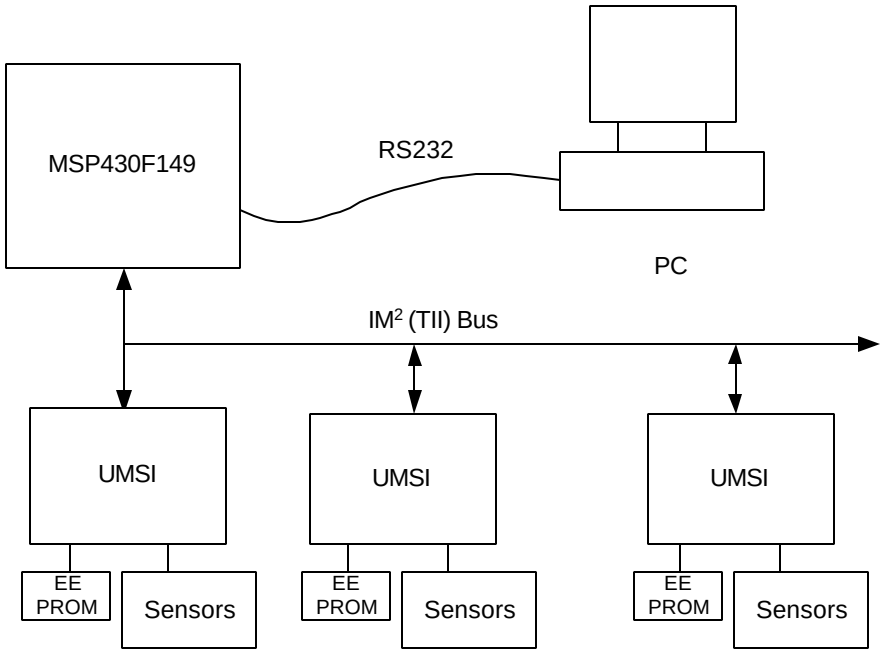


Figure 4.1: Diagram of IC Test System

MSP430F149 microcontroller was chosen to be the Controller in the IC Test System. Its work functions include: creating the IM^2 bus interface, capturing analog signals, converting them to digital values; responding the interrupt request; processing and transmitting the data between microcontroller and a host PC system.

4.3. Software Design

Software includes two parts: the software of MSP430F149 and the software of PC. The main tasks of PC software are: initial the communication port (RS232), receive the input command from user, send the command to MSP430F149 via RS232, wait the acknowledgement from the MSP430F149 (MSP430F149 receives the command), receive the data from MSP430F149, send the acknowledgement to the MSP430F149 (data have been received by PC). Figure 4.2 is the software data flow diagram.

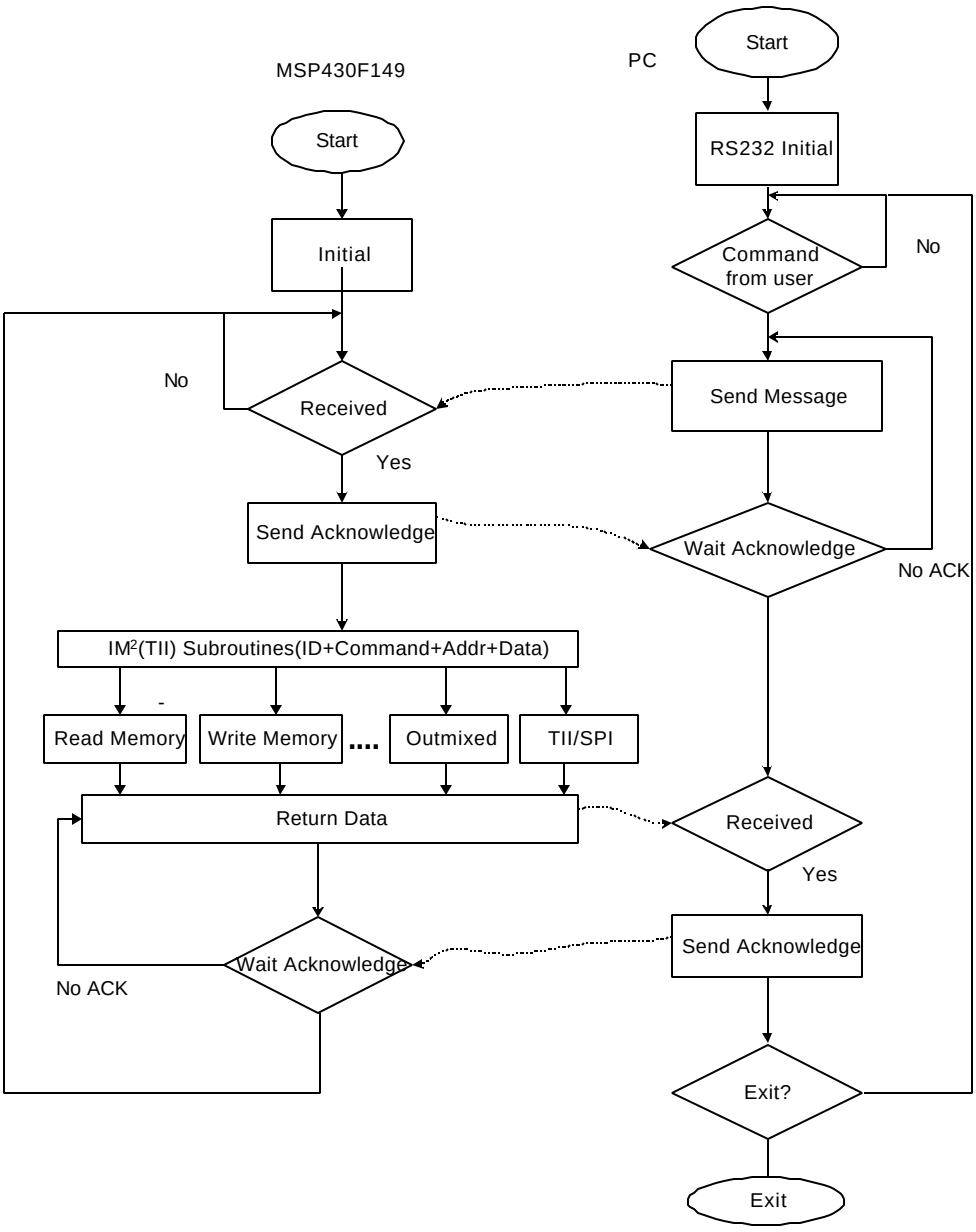


Figure 4.2: Data Flow Diagram of Softwares

The main tasks of MSP430F149 software are: initialize the communication port (RS232), wait the input command from PC, send the acknowledgement to the PC to indicate the command has been received, do IM^2 Bus Subroutines (ID+Command+Addr+Data), return the data to PC, wait the PC Acknowledgement.

The following subroutines of MSP430 are the basic drivers of MSP430 hardware.

High (port,number): Let the port.number pin High

Low (port,number): Let the port.number pin Low

Clock (port,number): Let the port.number send out exact continual clock signal. Use hardware timer of MSP430.

Gate (port,number,time): let the port.number send out a exact time gate signal. Using hardware timer of MSP430.

Send (data): Send the data to PC via RS232

Read (): Read the data from PC via RS232

AD (): sample the analog signal

Based on the basic software subroutines, the high level subroutines have been given in table 5.1.

Table 4.1 High level subroutines for MSP430F149.

Subroutines	**Functions**
Read_Memory(ID,Addr)	Read data from on-chip memory
Write_Memory(ID,Addr,Data)	Write data to on-chip memory
Read_IO(ID)	Read IO port
Write_IO (ID,Data)	Write data to IO port
LoadIDfromIO(ID)	Load chip ID from wired bond IO pins
LoadIDfromEEPROM(ID)	Load chip ID from TEDS (EEPROM)
OutMixed(ID,channel)	Enable the mixed signal sent out via DOUT
ReadSensor(ID,channel)	Readout the on-chip digital sensor (only temperature sensor on prototype chip)
TII_SPI_Switch(ID)	Switch the IM^2(TII) Bus with SPI Bus

Chapter 5

Conclusions

The existing fragmented sensor market is seeking ways to build low-cost, networked smart sensors. Many sensor network or fieldbus implementations are currently available, each with its own strengths and weaknesses for a specific application class. IEEE P1451.2 provides a smart transducer interface standard that isolates the choice of transducers from the choice of networks. This would relieve the burden of the manufactures from supporting sensor products across various networks, and would help to preserve the user's investment if it becomes necessary to migrate to a different network standard. However there are many limitations to the use of this standard, especially for the growing market of microsystems capable of reading multiple parameters while maintaining small size and low power dissipation. For these applications, IEEE P1451.2 is not adequate and other solutions must be sought.

In this research, an improved sensor communication bus has been introduced. The IM^2 Bus, a modified superset of the IEEE P1451.2 TII Bus, has been carefully defined**.** The design concept of this bus has been implemented in the design of the IM^2 Bus Interface Circuit which is one element in a Universal Micro-Sensor Interface circuit. In addition, a test system for interfacing to the UMSI circuit via the IM^2 bus has been implemented. This test system provides both a vehicle for testing the circuit as well as a method for qualification of the IM^2 bus. We are confident that this design provides not only a thorough research effort but also a product that can be useful in the sensor industry.

For future work, it is noted that the bus protocol as implemented on the UMSI chip could be more closely matched to the IEEE standard. An external EEPROM has been used in this project as TEDS. Although this method simplifies the interface circuit (UMSI) and helps to reduce design cost, in order to maintain consistency with IEEE P1451.2 protocol an internal, on-chip, EEPROM could be implemented. This would provide with uniform address coding better matched to the protocol of the IEEE P1451.2 standard and possibly more generally useful to the sensor industry.

REFERENCES

[1]. IEEE Instrumentation and Measurement Society, "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," *IEEE Std 1451.2*, 1997.

[2]. A. Mason, "Portable Wireless Multi-Sensor Microsystems for Environmental Monitoring", *Ph.D Thesis*, The University of Michigan, August 2000.

[3]. A. Mason, N. Yazdi, A. V. Chavan, K. Najafi, K. D. Wise, "A Generic Multielement Microsystem for Portable Wireless Applications," (Invited) *Proc. IEEE*, vol. 86(8), pp. 1733-1746, August 1998.

[4]. N. Yazdi, A. Mason, K. Najafi, and K. Wise, "A Low-Power Generic Interface Circuit for Capacitive Sensors," *Digest, Solid-State Sensor and Actuator Workshop*, Hilton Head Island SC, pp. 215-218, June 1996.

[5]. A. V. Chavan, A. Mason, U. Kang, and K. D. Wise, "A Programmable Mixed-Voltage Sensor Readout Circuit and Bus Interface with Built-In Self-test," *Digest*, *Int. Solid State Circ. Conf.*, San Francisco CA, pp. 136-137, February 1999.

[6]. Tom Cantrell, "Car 1451, Where are you? A Look at the IEEE 1451 Standard," *Circuit Cellar INK, Issue 103*, pp.78-82, February 1999.

[7]. Robert N. Johnson, "Building Plug-and-Play Networked Smart Transducers," *SENSORS*, P40-51, October 1997.

[8]. Tim Cummins, Dara Brannick, " An IEEE 1451 Standard Transducer Interface Chip With 12-b ADC, Two 12-b DAC's, 10-kB, Flash EEPROM, and 8-b Microcontroller," *IEEE Journal of Solid-State Circuit*, pp2112-2120, December 1998.

[9]. Lee H. Eccles, "A Brief Description of IEEE P1451.2," *Proc. Sensors Expro*, pp81-90, October 1997.

[10]. Richard L. Fischer, Jeff Burch, "The PICmicro® MCU as an IEEE 1451.2 Compatible Smart Transducer Interface Module (STIM)," *Microchip AN219*.

[11]. Analog Devices Company, "The ADuC812 MicroConverter as an IEEE 1451.2 Compatible Smart Transducer Interface," *MicroConverter Technical Note-uC003*.